2018

# AZURE SQL Database

ESTEBAN COTO ALFARO
JOHAN TORRES
DANNY XIE LI

## Introducción

Actualmente una de las estructuras de almacenamiento más usadas son la base de datos SQL relacional, este es un conjunto de datos pertenecientes a un mismo contexto que cumple con el modelo relacional. Actualmente la tecnología de computación en la nube ha crecido con los años, Microsoft ofrece una base de datos SQL en la nube por medio de los servicios de Azure, en la cual posee características como escalabilidad automática e ilimitada, despliegue no asistido, crece como sea necesario, paga por lo que usas, que más adelante se explicará las características que posee el servicio Azure con respecto a base de datos SQL en la nube, además de cómo configurar el firewall, como crear una instancia de la base de datos y entre otros temas que se desarrollaran.

En este artículo se explicará algunas consideraciones de diseño que debe tener el desarrollador a la hora de construir la aplicación usando la base de datos Azure en la nube, para que este tenga un mejor rendimiento del sistema. Además, se desarrollará el tema de sincronización de datos con la base de datos SQL y algunos patrones de diseño.

Uno de los puntos más relevantes en el mundo cibernético de la actualidad es el tema de la seguridad y el cuido que se le dé a los datos, principalmente a los más sensibles donde destacan contraseñas, números de tarjeta, número de seguro social (principalmente en E.E.U.U.). Se explica la categorización de la seguridad bajo el modelo CIA, uno de los más aceptados.

Realizar la conexión entre la base de datos y el lenguaje que se esté utilizando es uno de los aspectos más importantes ya que es por donde se le indica a la base de datos como accionar y sobre que aspectos actuar. Se explica cómo realizar la conexión mediante la ejecución de ejemplos de ADO.NET y ODBC, y al final la explicación de los servicios de datos WCF.

Microsoft además provee al usuario las herramientas necesarias para administrar la seguridad de las bases de datos dentro de Azure, así como los tipos de replicación y las auditorías para proteger la información frente a cualquier ataque externo que podría dejar consecuencias catastróficas para la base de datos. El capítulo 9 del libro de Pranab Mazumdar detalla dichas características a profundidad, así como la forma de uso respectiva.

## Capítulo 1 Empezando con la base de datos SQL

### Introducción a la computación en la nube

Algunas características de lo que ofrece la computación en la nube comparado con los servicios de hostings tradicionales:

- Escalabilidad automática e ilimitada: Si su servicio necesita más recursos, se aprovisionarán más recursos automáticamente o con un esfuerzo limitado.
- Despliegue no asistido: Si necesita implementar servicios o bases de datos adicionales, no ocupa realizar nada.
- Conmutación por falla incorporada: SI un servidor falla esto no se notará.
- Crecer como sea necesario; paga por lo que usas: Sólo paga por los recursos que utilizas.

### Typical Cloud Services

La computación en nube posee 3 tipos:

- SaaS (Software as a service): Esta plataforma suele ser en forma de aplicaciones web que están disponibles en Internet por un monto. Ejemplo son Google Apps.
- PaaS (Platform as a service): Este servicio ofrece una plataforma informática que facilita el uso e implementación de otros servicios, posee escalabilidad. Un ejemplo Amazon S3.
- IaaS (Infrastructure as a service): Este ofrece la infraestructura necesaria que ofrece escalabilidad asociada con la computación en la nube, como Windows.

### Descubriendo la plataforma de Microsoft Azure

Este posee tres componentes, para poder proveer una lista de servicios completos necesarios para construir una escalabilidad alta y soluciones seguras:

- Windows Azure: Es una colección virtual de Sistemas operativos Microsoft que puede correr aplicaciones web y servicios en la nube.
- Cloud services: Conjunto de servicios que provee capacidades básicas.
- SQL Database: Base de datos transaccional de Microsoft ofrece computación en la nube en Microsoft SQL Server 2012.

### Microsoft Azure

El ambiente de Microsoft Azure permite múltiples escenarios de negocios para florecer, estos incluyen:

- Seasonal aplications: Desarrollar sitios web o servicios que tiene una tendencia para crecer y contrata a través del tiempo provee oportunidades.

- Short life span: El desarrollo de prototipos o aplicaciones con vidas cortas es atractivo, como sitios de registros de eventos.
- Split storage: Ciertas aplicaciones necesitan guardar sus registros en un lugar seguro, pero no requiere acceso frecuente o requiere de una disponibilidad alta.
- Small companies and ISVs: Las compañías pequeñas que no pueden permitirse una infraestructura grande y compleja para comenzar su negocio puede aprovechar de los recursos de Microsoft Azure.

**Geographic location**

Mejorar el rendimiento escogiendo el geolocation correcto. Cuando se selecciona la localización geográfica se considera lo siguiente:

- Rendimiento: Cuando los datos están cerca de los usuarios la latencia de la red puede no sea notable este mejora la experiencia del usuario.
- Recuperación de desastre: Si la disponibilidad de la plataforma de la nube es importante debe dispersar los servicios y datos en múltiples regiones.
- Legal factors: Los datos que se almacenan en la nube, y asegurarse de que no esté ligado por regulaciones especificados.

**Storing Data in Azure**


Almacén de Windows Azure puede ser accesado directamente desde un ambiente corporado usando llamadas HTTP/s, este puede hacer request directamente a la instancia de la base de datos SQL usando ADO.NET o ODBC, porque la base de datos SQL soporta la Tabular Data Stream (TDS) protocolo que el servidor de SQL usa.

Existen 4 tipos de almacenamiento:

- Windows Azure storage
- Table: Almacena pares de llave-valor
- Blobs: Una interfaz para almacenar archivos con un límite máximo de 200GB o 1TB de almacenamiento, este se puede accesar usando request HTTP a través de llamadas REST.
- Queue: Es un mecanismo para almacenar mensajes que son consumidos por otros aplicaciones o servicios. Un uso típico es enviar mensajes XML, además este puede accesarse a través de REST.
- SQL Database: Es una base transaccional que provee acceso familiar a los datos a través de ADO.NET u otros proveedores que la habilidad de manipular datos usando el estándar T-SQL declaración.

*Table 1-1. Storage Summary in Azure*

| Storage Mode | Maximum Size | Access | Format | Relational |
|---|---|---|---|---|
| Table | N/A | ADO.NETREST | Rows and columns | No |
| Page Blob | 1TB | REST | File | No |
| Block Blob | 200GB | REST | File | No |
| Queue | 64KB* | REST | String | No |
| SQL Database | 150GB | ADO.NET | Rows and columns | Yes |

** Recommended limit*

## Creando la instancia de la base SQL

Primero debe crear un servidor de base de datos SQL, la base de datos maestro se aprovisiona automáticamente. Esta base es solo lectura y contiene configuración e información de seguridad para su base de datos. Puede usar el Portal de administración de Windows Azure o declaraciones T-SQL contra la base de datos maestro usando el Management studio del servidor SQL.

## Usando el comando T-SQL

Creando una nueva base de datos usando T-SQL, para esto debe conectarse con la base de datos maestro para crear nuevas bases. Para crear una base de datos nueva usando el administrador de SQL server, login usando la cuenta administradora y correr el siguiente comando T-SQL:

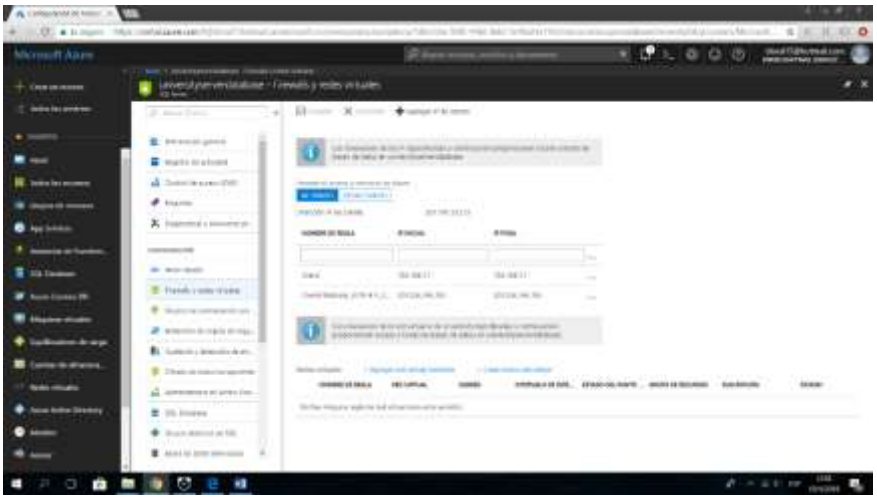
```
CREATE DATABASE TestDB (MAXSIZE = 10 GB)
```

Si no se le indica el `MAXSIZE` por defecto es `1 GB`.

## Configurando el Firewall

La base de datos SQL implementa firewall. Esto es un beneficio para proteger la base de datos, una regla por defecto de Firewall es que nadie puede conectarse a un nuevamente creada un servidor de base de datos SQL. Pasos a seguir:

a. En el portal de administración de Windows Azure, seleccionar la pestaña de base de datos SQL en el panel de la parte izquierda.
b. Selecciona la pestaña de Firewalls y redes virtuales.
c. Selecciona el nombre del servidor que quiere agregar la regla de firewall.
d. Selecciona la pestaña de configuración.
e. En la sesión de Permitir el acceso a servicios de Azure, ingrese el nombre de la regla y la dirección de IP inicial y el IP final.



f. Si tiene servicios de Windows Azure que necesita acceso al servidor de la base de datos SQL, selecciona la opción donde se encuentra la sección Permitir el acceso a servicios de Azure.

Se puede ver y editar configuración de Firewall usando directamente T-SQL, conectándose a la base de datos maestro con la cuenta administrador y usando los siguientes objetos:

- `Sys.firewall_rules`
- `Sp_set_firewall_rule`
- `Sp_delete_firewall_rule`

### Conectando con SQL Server Management Studio

Siga los siguientes pasos para conectar con la instancia de la base de datos SQL usando el SQL Server Management Studio:

1. Debe obtener el nombre del servidor de la base de datos SQL Server.
2. Empezar el SQL Server Mangement Studio. Haga click en cancelar en la pantalla de Login.
3. Haga click en el botón New Query, se abre la ventana de Login, en este digite los siguientes datos:
    - Nombre del servidor
    - Autentificación: Selecciona autentificación del servidor SQL
    - Login: Escriba el nombre de usuario del administrador.
    - Password: La contraseña del usuario administrador.
4. Por defecto al tocar click en Conectar autentifica a la base de datos maestro.

### Creando usuarios y Logins

Cuando está conectado se crea el Login usando el comando `CREATE LOGIN`. Luego necesitas crear una cuenta de usuario en la base de datos y asignarle permisos a esa cuenta.

### Creando un nuevo Login

Conectarse a la base de datos maestro usando la cuenta administradora y ejecutar el siguiente comando:

```
CREATE LOGIN test WITH PASSWORD = "T3stPwd001"
```

Este no se puede registrarse hasta que se haya creado un usuario, para verificar que el login se haya creado correctamente ejecutar el siguiente comando:

```
SELECT * FROM sys.sql_logins
```

Si intenta crear la cuenta login en una base de datos usuario, usted recibe un error debido a que el login debe crearse en una base de datos maestra.

### Creando un nuevo usuario

Conectar a la base de datos del usuario usando la cuenta del administrador y ejecute el siguiente comando:

```
CREATE USER test FROM LOGIN test
```

Si intenta crear un usuario sin la creación de una cuenta login recibe un error.

### Asignando a derechos de acceso

Cuando ya hayas creado el login y la cuenta usuaria. Pero esta cuenta usuaria no se le ha asignado ningún derecho de acceso. Para permitir que la cuenta test tenga acceso ilimitado a la base de datos del usuario, se debe asignarle el usuario al grupo `db_owner`:

```
EXEC sp_addrolemember 'db_owner', 'test'
```

En este punto ya está listo para usar la cuenta test para crear tablas, vistas, procedimientos almacenados y más.

### Comprendiendo la importancia de la base de datos SQL

La base de datos SQL, es un modelo de pay-as-you-go, para observar la banda ancha de consumo y la base de datos que provisionó desde un punto de vista de facturación ejecute los siguientes comandos:

```
SELECT * FROM sys.database_usage    --databases defined

SELECT * FROM sys.bandwidth_usage  --bandwidth
```

El primer comando retorna el número de base de datos disponibles por día de un tipo específico ya sea web o negocio. El segundo comando muestra el desglose de consumo por hora de la base de datos.

### Limitaciones en la base de datos SQL

Crear base de datos y usuarios requiere de un manual se scripts y cambiando de conexiones de la base de datos. La diferencia entre SQL Server y la base de datos SQL yace en un conjunto de diseños básicos de principios de computación en la nube, de la cual el rendimiento, facilidad de uso, y la escalabilidad.

### Security

Para empezar por seguridad se debe seguir las siguientes restricciones:

- **Encriptación**: Aunque la base de datos SQL usa SSL para la transferencia de datos, este no soporta funciones de encriptación de datos disponibles en SQL Server. La base de datos provee un soporte para funciones hash.
- **SSPI autentificación**: La base de datos solo soporta logins. Como resultado, la red logins que usa Security Support Provider Interface (SSPI) no son soportados.
- **Restricciones de conexión**: en ciertos casos, la conexión de la base de datos se cierra por una de las siguientes razones:
    - Uso exceso de recursos
    - Larga-ejecución de query
    - Larga-ejecución de una simple transacción
    - Conexión Idle
    - Failover debido a una falla de servidor.
- **Nombre de usuarios no permitidos**: Por razones de seguridad no se permiten los siguientes nombres de usuario: sa, admin, administrator, guest, root.
- **Login name:** En ciertos casos debes adjuntar el nombre del servidor al login name para ingresar correctamente en el siguiente formato: [loginName] @ [servername]
- **TCP port 1433**: Sólo el puerto 1433 es permitido. No es posible definir otro puerto para la base de datos SQL.

### Resguardos

Backup/restore operations, clonar operaciones (clonar una base de datos a otra usando el comando `CREATE DATABASE`), archivos logs (no puede accesar estos archivos tampoco ni crear copia de estos archivos), exportar e importar datos (se usa las características `import y export`), `restore`(se puede usar la característica del punto de restauración en el tiempo).

### Objetos

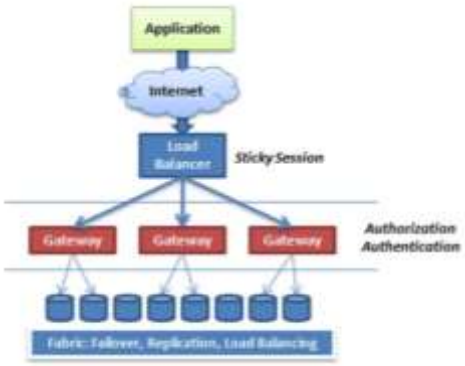
Algunos objetos se encuentran disponible en SQL Server que no están disponibles en la base de datos SQL. Algunas limitaciones son:

- `CLR`
- System functions como `OPENQUERY`, `OPENXML`, `OPENROWSET`, `OPENDATASOURCE`
- System stored procedures: solo un pequeño subsistema de procedimientos almacenados está disponible en SQL como las siguientes categorías: Catálogo, Database engine, Security stored procedures.
- System tables
- System views: algunos views disponibles son `sys.sql_logins`, `sys.views`, `sys.databases`, `sys.columns`, `sys.objects`.
- Heap tables: SQL no permite el uso de tablas Heap, todas las tablas deben tener un índice agrupado.

## Capítulo 2 Consideraciones de diseño

La plataforma Azure ofrece 4 distintos modelos de almacenamientos: Blob, Tabla, Queue, y base de datos SQL.

### Alta disponibilidad




Cuando se diseña aplicaciones, los desarrolladores de software y arquitectos se preocupan de la de los requerimientos de alta disponibilidad. La base de datos SQL usa una topología muy elaborado que maximiza el trabajo de redistribución, transparencia y recuperación. En la siguiente imagen enseña una alta implementación de la base de datos SQL, de cómo es debe ser de avanzado la estructura de backend.

### Throttling

La base de datos SQL se ejecuta en un entorno multiusuario, esto quiere decir que las instancias de su base de datos comparten recursos del servidor con bases de datos y otras compañías. Si la aplicación emite una consulta grande que afecte a otras bases esta conexión finaliza, lo desconecta automáticamente.

Las siguientes condiciones terminaran la conexión de la base de datos:

- Lock Consumption: Si tu aplicación consume más de 1 millón de locks, este recibirá el código de error 40550.
- Uncommitted Transactions: Una transacción bloqueada interno de los recursos por más de 20 segundos termina con el error de código 40549.
- Log File Size: Si el archivo log para una sola transacción excede 1GB en tamaño, esta conexión termina con el error 50552.
- TempDB: SI se corre largas transacciones, o conjuntos grandes de comandos, o largo operaciones de ordenamiento que consume más de 5GB de espacio, la sesión termina con el error 40551.
- Memory: Si sus declaraciones consumen más de 16 MB de memoria por 20 segundis, su sesión será terminada con el código 40553
- Database size: Si se excede en su configuración máxima de tamaño, cualquier intento de insertar o actualizar datos tira el error 40544.
- Idle Connections: Alguna conexión que este más de 30 minutos será terminada, no retorna error.
- Transactions: Transacciones que perduran más de horas serán terminadas con el error 40549.
- Denia lof service Attacks: Si realiza muchos intentos de login desde de un especifico IP address, cualquier intento para conectar desde esa IP address fallara por un periodo de tiempo para proteger la base de datos.

- Network problems: Si hay un error de red en el origen de una sesión terminada, no recibirá un error de código.
- Failover: Cuando una base de datos SQL está en el medio de una caída de su base de datos sobre otro node, sus sesiones activas serán desconectadas.
- High Activity: Si el servidor hosting de su base de datos experimenta un estrés significante o excede su limites operativos, la base de datos podría desconectarse de las sesiones con el error 40501.

Otras condicionales adicionales, llamadas errores transitorios, podría causar que la sesión termine.

- Error 20: La instancia de la base de datos que intenta conectar o soporta encriptación.
- Error 64: La conexión se estableció satisfactoriamente con el servidor, pero ocurrió un error durante el proceso de login.
- Error 233: El cliente no pudo establecer una conexión porque hubo un error durante la inicialización de un proceso antes de un login.
- Provider: TCP Provider, error: 0 – An existing connection was forcibly closed by the remote host.
- Error 10053: Un error de nivel-transporte ha ocurrido cuando se estaba recibiendo resultados desde el servidor.
- Error 10054: Un error de nivel-transporte ha ocurrido cuando se está enviando una solicitud al servidor.
- Error 10060: La red-relacionado o la instancia-especifica error ocurrido mientras se establecía una conexión con el servidor SQL.

**Consideraciones de diseño de la aplicación**

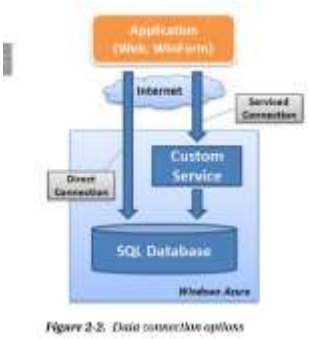
Cuando se considera de como diseñar una aplicación hay que tomar la ventaja que ofrece la base de datos SQL, hay que evaluar los siguientes puntos:

- Database roundtrips: Cuantos roundtrips son necesarios para que se ejecute una función en especifica en una aplicación.
- Caching: Puede mejorar el tiempo de respuesta almacenando en caché recursos en la máquina del cliente o almacenando datos temporales más cerca del consumidor.
- Propiedad de carga lenta: Para reducir los roundtrips es mejor solo cargar los datos que son necesarios o requeridas por las funciones.
- Interfaces de usuarios asíncronos: Multi hilos puede asistir en proveer aplicaciones más receptivas.
- Shards: Es una forma de dividir los datos en base de datos múltiples que de manera sea lo más transparente posible al código de su aplicación.

**Sincronización de datos**

Hay dos maneras para sincronizar datos con SQL Database: Microsoft Sync Framework y el SQL Data Sync service.

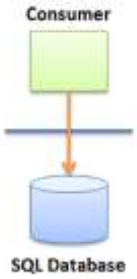
## Directo vs conexiones de servicio




Desarrollar servicios Azure para mantener la conexión de la base de datos en una red local, y enviar los datos devuelta al cliente usando SOAP o mensajes REST. Una conexión directa puede ser establecida con una base de datos desde la aplicación, en caso de que la aplicación emita declaraciones T-SQL para recuperar datos.

## Patrones de diseño

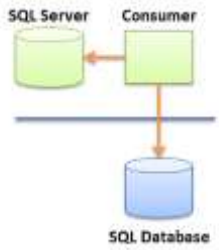
## Conexión directa



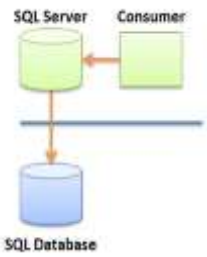

El siguiente patrón puede ser la forma más simple de conectividad a una instancia de SQL Database. El consumidor puede ser una aplicación ubicada en una red de una corporación o un servicio de Windows Azure que se conecta directamente a la instancia de la base de datos.

## Smart Branching

Este patrón describe una aplicación que contiene suficiente lógica para determinar si el dato necesita cargar se encuentra localizado en la nube o en una base de datos local.




## Ramificación transparente




Este patrón depende del consumidor para determinar si los datos son locales o en la nube, este patrón elimina esta preocupación al consumidor. Este patrón está diseñado para implementarse en aplicaciones que son difíciles de modificar o que los costos de implementación son prohibidos.

## Sharding

Sharding múltiples base de datos pueden ser accesado simultáneamente en un estilo de lectura y escritura, que puede localizarse en ambientes mixtos es decir local o en la nube.

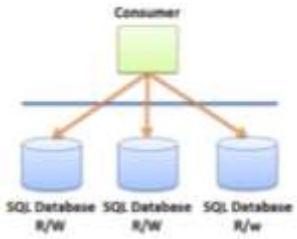

## Read-Only Shards

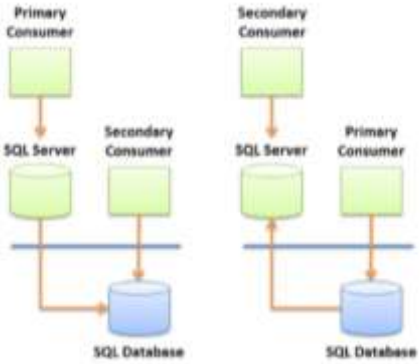

Figure 2-8. Read-only shard topology

Shards puede ser implementado en múltiples formas. Por ejemplo, se puede crear un read-only shard (ROS). Aunque el shard es alimentado desde una base de datos que acepta operaciones de lecturas/escritura, sus registros son solo lectura para los consumidores.

## Read-Write Shards

En un read-write Shard (RWS), todas las bases de datos son considerados lectura/escritura. En este caso, usted no necesita usar la topología de replicación que usa el SQL Data Sync Framework, porque es una simple copia de cada registro dentro de un shard.




## Offloading

En el patrón de offloading, el consumidor primario representa una aplicación de sitio existente con su propio base de datos; pero un subconjunto de datos (o su base de datos entera) es replicado a la nube de la base de datos usando SQL Data Sync. Se puede accesar de dos formas

## Aggregation

En su forma simple, el patrón aggregation provee un mecanismo para colectar datos desde múltiples proveedores de datos dentro de una instancia de SQL Database.

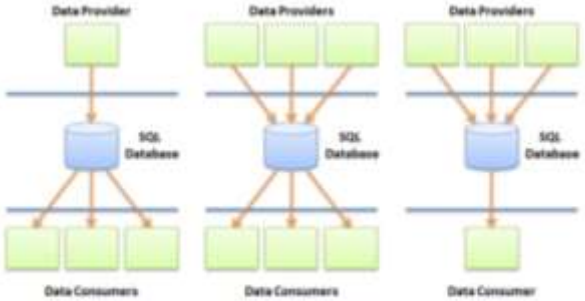


*Figure 2-11. Aggregation patterns*

## Mirroring

*Figure 2-12. Mirror pattern*

Este patrón es una variación del patrón offloading donde el consumidor secundario puede ser una entidad externa. En adición, este patrón implica que two-ways replication existe, entonces esto cambia en cualquier base de datos que son replicados en otra base de datos.

## Combinando patrones

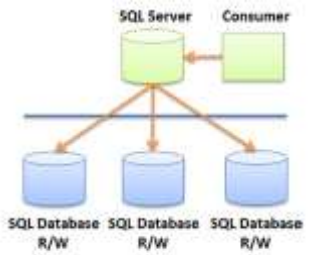


*Figure 2-13. Transparent branching + RWS patterns*

### Transparent Branching + RWS

Este patrón puede ser usado para descarga, dentro de la nube, almacenar una historia de datos que un existente Enterprise Resource Planning (ERP) que genera la aplicación.

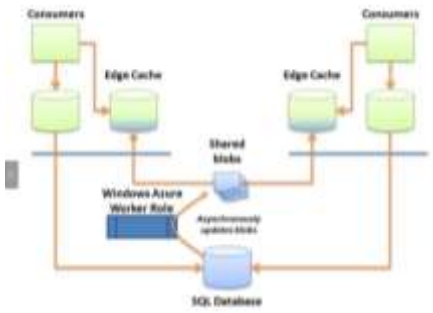
## Agregación en cascada

Es aplicado para generar un resumen de la base de datos. El mecanismo para copiar datos desde una instancia SQL Database a otra debe ser realizado usando procesos de high-level, tal como el proceso de trabajo Windows Azure.

## Otras consideraciones

### Blob Data Stores

Blobs son archivos que pueden ser almacenados en Windows Azure. Estos pueden ser accesado a través de REST. Los Blobs pueden ser usados como backup y mecanismo de transferencia entre consumidores.




### Edge Data Caching

Se puede almacenar tablas relativamente estáticas en la memoria, guardarlas como Blobs (forma de almacenamiento en memoria) para que otros sistemas de almacenamiento en caché utilicen el mismo caché y crear un mecanismo para actualizar su caché de datos utilizando colas en Azure.

## Capítulo 3: Seguridad

A manera introductoria al tema seguridad, uno de los puntos más sensibles en la actualidad, este está conformado por términos básicos tales como: confidencialidad, integridad y disponibilidad conocido como CIA (por sus siglas en inglés). CIA es una de las formas más aceptadas de categorización de la seguridad.

Confidencialidad: capacidad de garantizar que solo usuarios autorizados puedan acceder a los datos. Se compone a su vez de ciertos conceptos:

- Encriptación: Crea un texto cifrado que se puede descifrar a través del uso de una clave compartida.
- Hashing: Genera un texto cifrado que no se puede descifrar (principalmente se implementa con el almacenamiento de contraseñas).
- Control de acceso: Controla el acceso de datos en función de la información contextual.
- Autenticación: Controles quien puede acceder a la base de datos y cuales objetos en la base de datos el usuario puede acceder.
- Firewall: Utiliza tecnologías para limitar la conectividad de red a una lista de máquinas conocidas.

Integridad: objetivo de garantizar que la información sea modificada solo por usuarios previamente autorizados, y que el daño se pueda deshacer en caso de ser necesario. La integridad de los datos puede verse vulnerada de distintas formas, como un “SQL Injection attack” o la ejecución involuntaria de la instrucción “TRUNCATE” en una tabla borrando todos los registros. Realizando una generalización sobre el tema, se podría realizar la implementación de unas medidas de integridad en las bases de datos, tales como:

- Autorización: controles de quine puede cambiar datos.
- Apoyo: crear un “snapshot” de la base de datos a partir del cual se pueden recuperar los datos.
- Acceso basado en roles: proporcionar los derechos mínimos de acceso a diferentes roles.

- Auditoría: seguimiento del acceso a la base de datos y los cambios en los datos para proporcionar una pista de auditoría para su análisis.

Disponibilidad: garantiza el tiempo de actividad del servicio en el que se puede acceder a los datos de ser necesario. El diseño de sistemas de una disponibilidad alta resulta ser muy complejo, algunas de las tecnologías involucradas en la alta disponibilidad son:

- Discos redundantes: permite la recuperación de pérdida de un "disk spindle", involucra la configuración "RAID" comúnmente.
- Redes Redundantes: permite sobrevivir a la pérdida de componentes de red como, tarjeta de red o un enrutador.
- Servicios redundantes: permite sobrevivir a la interrupción de servicios como, seguridad y bases de datos.
- Hardware redundante: permite sobrevivir a la pérdida de hardware de una máquina, ya sea CPU o un chip de memoria.
- Escalabilidad: entrega información a una velocidad constante bajo carga.
- Prevención de DoS (Denia lof Service): evita de forma exitosa ataques DoS que, a su vez, previene la disponibilidad de datos.

Para tener una implementación correcta de la seguridad en los datos se deben detectar aquellas columnas en la base de datos que contengan información sensible, comúnmente se implementa en contraseñas o números de cuentas bancarias.

Tomando el ejemplo presente en el libro para la implementación de varios métodos tales como el de cifrado, hash y otros, tendría un diagrama como se muestra a continuación.

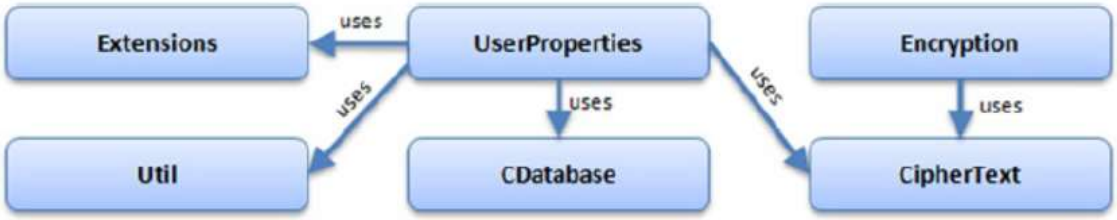



*Figure 3-2. Object model used in the examples*

Cifrado: la base de datos SQL requiere el uso de encriptación SSL para la comunicación, eso implica que su sensibilidad los datos se transmiten siempre de forma segura entre sus clientes y su instancia de la base de datos SQL.

Hashing: dentro de las fortalezas del hashing es el de poder modificar los datos y ser capaz de detectar si un cambio es realizado sin autorización.

Certificado: el certificado implica que se está encriptando con dos claves, público/privada. La clave pública se utiliza para cifrar los datos, mientras que la privada se usa para descifrar los datos.

Después de la aplicación de los tres métodos previos, pasa a la etapa de control de acceso donde se divide en dos subcategorías, autenticación y autorización.

Autenticación: proceso que verifica si se es quien se dice ser.

Autorización: da la capacidad de controlar quién puede realizar acciones, después de ser autenticado verifica qué acciones puede realizar.

## Capítulo 5: Programación con base de datos SQL

Este capítulo inicia mencionando aquellos aspectos previos que fueron explicados en los capítulos anteriores para posteriormente explicar la conexión a la base de datos SQL. Microsoft facilita la conexión de una aplicación con la base de datos SQL dado que proporciona conexión para varias bibliotecas de clientes como ADO.NET, Java y PHP. Tal y como lo menciona el título de este capítulo se expondrá en el resumen código ilustrativo para comprender no solo de que trata cada punto si no también como poder realizar su implementación.

ADO.NET: para la implementación de ADO.NET se deben de seguir una serie de pasos.

- Realizar la conexión: este consta en pasos a seguir.
    1. Crear una clase conexión: `using System.Data.SqlClient;`
    2. Se agrega un método para obtener un string de conexión:

```
string GetConString()
{
   return
   "Server=server;Database=AdventureWorks2012;UserID=sa;Pas
   sword=password;";
}
```

    3. En el "click event" del botón se debe agregar el siguiente código, el cual llamará al método creado anteriormente (punto 2). Este devuelve el string de conexión, establece y crea una conexión a la base de datos local y luego cierra la conexión:

```
private void button1_Click(object sender, EventArgs e)
{
      string connStr = GetConString();
      using (SqlConnection conn = new
SqlConnection(connStr))
      {
            try
            {
                  conn.Open();
                  MessageBox.Show("Connection made.");
            }
            catch (SqlException ex)
            {
                  MessageBox.Show(ex.Message.ToString());
            }
            finally
            {
                  conn.Close();
            }
```

```
        }
    }
```

4. Al correr la aplicación si dice “Connection made.” significa que fue creada de forma correcta.

5. En el método creado en el punto 2 se le debe realizar un cambio de forma que quede como lo siguiente:

```
string GetConString()
{
      return
      "Server=tcp:servername.database.windows.net;Database=
      AdventureWorks2012;
      UserID=username@servername;Password=password;
      Trusted_Connection=False;Encrypt=True;Connection
Timeout=30";
}
```

6. Antes de ejecutar nuevamente la aplicación, esta vez con el cambio realizado en el punto anterior (punto 5) se debe asegurar que la configuración del firewall de Azure esté actualizada, esto se puede ver a través de la página de administración de la base de datos SQL. La configuración esta correcta si dice “Connection made.” tal como se menciona en el punto 4.

- Lector de datos: a continuación, se modificará el código del “click event” que será explicado en los siguientes pasos.
    1. En el “click event” se agregará código que utilizará la clase SqlDataReader para ejecutar el comando SELECT hacia la base de datos SQL e iterar sobre SqlDataReader.

```
private void button1_Click(object sender, EventArgs e)
{
      string connStr = GetConString();
      using (SqlConnection conn = new
      SqlConnection(connStr))
      {
            SqlCommand cmd = new SqlCommand("SELECT
            FirstName, LastName FROM Person.Person",
            conn);
            conn.Open();
            SqlDataReader rdr = cmd.ExecuteReader();
            try
            {
                  while (rdr.Read())
                  {
                        listBox1.Items.Add(rdr[0].ToString());
                  }
                  rdr.Close();
            }
            catch (SqlException ex)
            {
                  MessageBox.Show(ex.Message.ToString());
            }
      }
```

```
}
```

2. Una vez agregado el código marcado en negrita en el punto 1, se procede a la ejecución donde según el ejemplo se obtendrán todos los nombres y apellidos de las personas que se encuentren en ese momento en la tabla "Person".

- Utilización del Dataset: como se mostró en el ejemplo anterior donde SqlDataReader utiliza prácticamente la misma sintxis que una consulta regular SQL, ahora se dispondrá de la utilización de las clases SqlCommand y SqlDataAdapter para realizar consultas SQL, se deben seguir los siguientes pasos:
    1. En el "click event" se debe reemplazar el código que ya existe por el siguiente:

```
using (SqlConnection conn = new SqlConnection(connStr))
{
      try
      {
            using (SqlCommand cmd = new SqlCommand())
            {
                  conn.Open();
                  SqlDataAdapter da = new SqlDataAdapter();
                  cmd.CommandText = "SELECT FirstName,
                  LastName FROM Person.Person";
                  cmd.Connection = conn;
                  cmd.CommandType = CommandType.Text;
                  da.SelectCommand = cmd;
                  DataSet ds = new DataSet("Person");
                  da.Fill(ds);
                  listBox1.DataSource = ds.Tables[0];
                  listBox1.DisplayMember = "FirstName";
            }
      }
      catch (SqlException ex)
      {
            MessageBox.Show(ex.Message.ToString());
      }
}
```

    2. Ejecutando la aplicación, esta nuevamente mostrará un cuadro con la lista de los nombres y apellidos de las personas en la tabla "Person".

ODBC: para verificar que la conexión con las clases ODBC funciona se siguen los siguientes pasos:

1. Se modificará el método GetConString de tal forma que lo que retorne sea un string de conexión ADO.NET o ODBC:

```
enum ConnType
{
      ADO_NET = 1,
      ODBC = 2
}
string GetConString(ConnType connType)
{
```

```
        if (connType == ConnType.ADO_NET)
              return
              "Server=tcp:servername.database.windows.net;Dat
              abase=AdventureWorks2012;
              User ID=username@servername;Password=password;
              Trusted_Connection=False;Encrypt=True;";
        else
              return "Driver={SQL Server Native Client

              10.0};Server=tcp:servername.database.windows.ne
              t;
              Database=AdventureWorks2012;Uid=username@server
              name;Pwd=password;Encrypt=yes;";
  }
```

2. Realice la siguiente declaración:
   ```
   using System.Data.Odbc;
   ```
3. En el "click event" al igual que con ADO.NET, pero en esta ocasión se utilizarán las clases ODBC, donde mediante la utilización del método GetConString del punto 1, se devolverá un string de conexión ODBC:
   ```
   string connStr = GetConString(ConnType.ODBC);
   using (OdbcConnection conn = new OdbcConnection(connStr))
   {
         try
         {
               conn.Open();
               OdbcDataAdapter da = new OdbcDataAdapter();
               OdbcCommand cmd = new OdbcCommand("SELECT
               FirstName, LastName FROM Person.
               Person", conn);
               cmd.CommandType = CommandType.Text;
               da.SelectCommand = cmd;
               DataSet ds = new DataSet("Person");
               da.Fill(ds);
               listBox1.DataSource = ds.Tables[0];
               dataGridView1.DataSource = ds.Tables[0];
               listBox1.DisplayMember = "FirstName";
         }
         catch (OdbcException ex)
         {
               MessageBox.Show(ex.Message.ToString());
         }
   }
   ```
4. Previo a realizar la ejecución del programa, en código del "click event" de ADO.NET se debe ingresar la siguiente línea al inicio:
   ```
   string connStr = GetConString(ConnType.ADO_NET );
   ```
5. Se procede a la ejecución del programa.

Se da la explicación de como se debe realizar con sqlcmd que su resultado es el mismo y se ejecuta mediante línea de comando.

Servicios de datos WCF: previamente se le conocía como servicio de datos ADO.NET. es la herramienta que permite la creación y el consumo de servicios OData, el Open Data Protocol, es un nuevo estándar de intercambio de datos que permite un mayor intercambio de datos entre diferentes sistemas.

- Crear un servicio de datos: la creación de un servicio de datos se explica de forma completa en el siguiente link: https://msdn.microsoft.com/es-es/library/cc668184.aspx, donde se utiliza Visual Studio, contiene tanto código C# como Visual Basic.
- Conexión del servicio al modelo: se necesita realizar la conexión entre el servicio de datos y el modelo de datos para que el servicio sepa de donde realizar la obtención de dato, para esto, si se siguió debidamente el link del punto anterior, se debe realizar el cambio del punto 1 al 2 en el código:

```
1. public class NorthwindCustomers : DataService
2. public class NorthwindCustomers : DataService<northwindEntities>
```

- Creando la aplicación del cliente: para la creación de la aplicación del cliente se puede asesar el siguiente link: https://msdn.microsoft.com/es-es/library/cc668184.aspx?cs-save-lang=1&cs-lang=csharp#Anchor_2. Donde se muestra tanto los pasos de la creación, así como el código para su implementación, igualmente que en el punto anterior este (código) viene en C# y Visual Basic.
- Agregar funciones de filtrado: En la siguiente dirección se encuentra una aplicación para filtración de datos, https://msdn.microsoft.com/es-es/library/cc668184.aspx?cs-save-lang=1&cs-lang=csharp#Anchor_3.

## Capítulo 8: Base de Datos Azure SQL

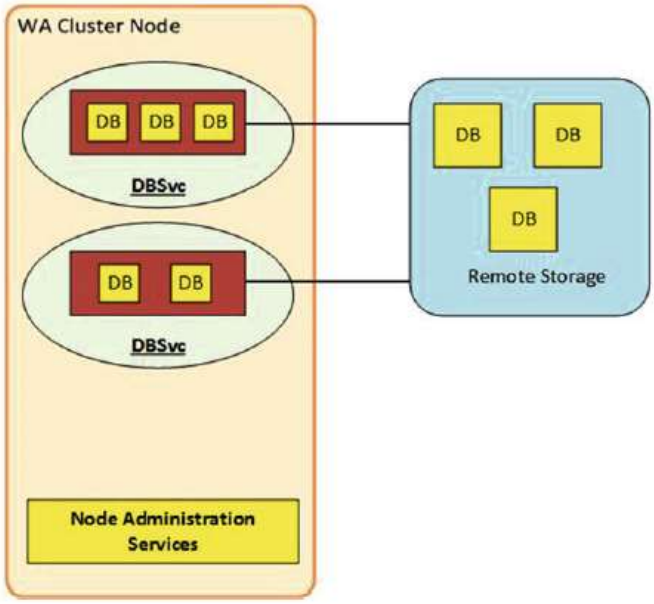
La base de datos Azure SQL es un servicio de bases de datos relacionales en Microsoft Azure, basados en el motor de Microsoft SQL Server, y con casi todas las capacidades de SQL Server para misión crítica. Azure SQL está diseñado para proveer bases de datos altamente disponibles y escalables como un servicio con rendimiento predecible, continuidad de negocios y capacidades de protección de datos.

### ARQUITECTURA DE LA BASE DE DATOS AZURE SQL

Azure SQL está construido sobre el framework de Windows Azure, el cual provee administración de máquina y la funcionalidad de aplicación distribuida. Un cluster de SQL Azure está constituido por un anillo de control y varios anillos "inquilinos", donde cada anillo equivale a un cluster físico de Windows Azure que consiste en una colección de nodos que ejecutan una o más aplicaciones.

### ANILLO INQUILINO

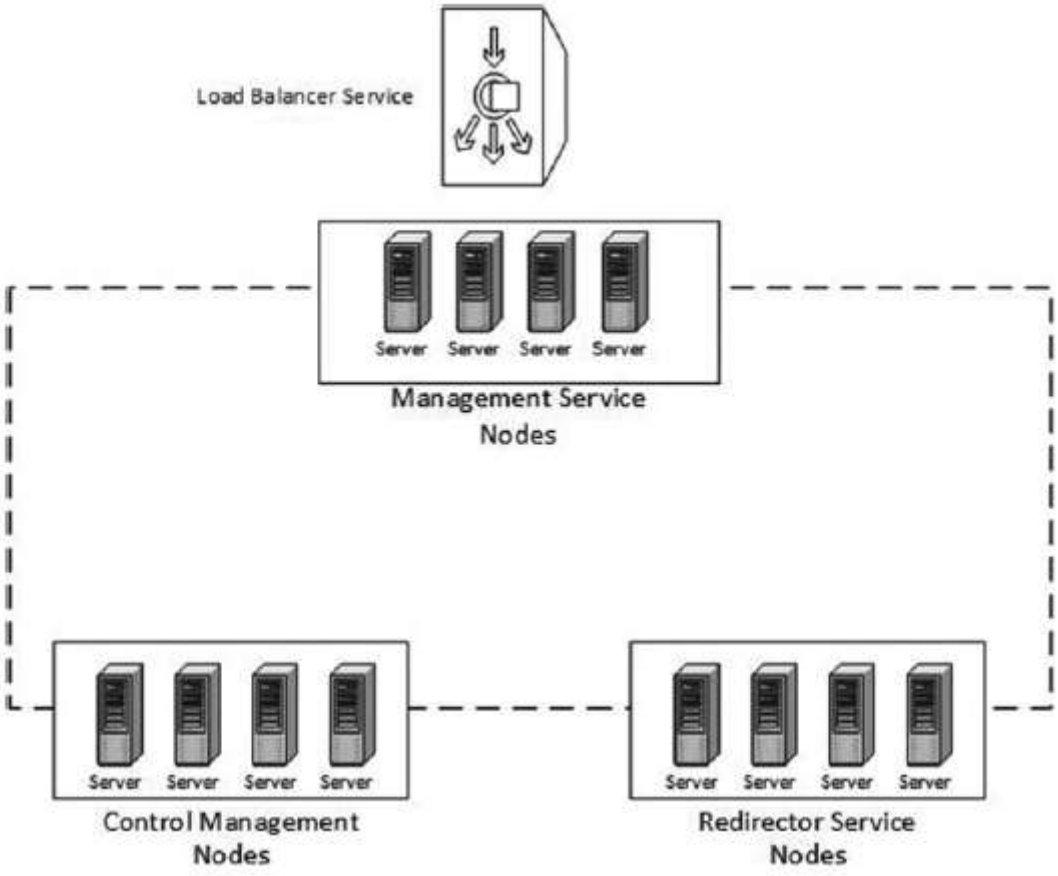
Un anillo inquilino es un cluster físico de Windows Azure, donde cada nodo del cluster está diseñado para ejecutar una aplicación. Dicha aplicación es de tipo DBService, el cual es un servicio del motor de SQL Server o del motor Hekaton (procesamiento de transacciones en línea dentro de la memoria).




La arquitectura del anillo inquilino permite la creación y el funcionamiento de múltiples DBServices, perteneciendo a uno o varios clientes. Ya que cada DBService es independiente de otras aplicaciones en DBService ejecutando dentro del mismo nodo en el anillo inquilino, esta arquitectura permite al mismo nodo alojar varias bases de datos cliente.

ANILLO DE CONTROL

El anillo de control provee los servicios de administración, provisionamiento y redireccionamiento dentro del cluster físico de Azure. Además, determina la ubicación de las bases de datos en los anillos inquilinos y el enrutado hacia las bases de datos inquilinas apropiadas.




NODO DE ADMINISTRACION DEL CONTROL

Provee los servicios de administración interna del cluster, los cuales proveen capacidades como administración de la capacidad, del anillo inquilino, migración, entre otros. Uno de los componentes clave del nodo de administración del control es el almacenamiento de metadatos del cluster, el cual es el punto de todos los metadatos relacionados con los clusters de Azure SQL (estado del cluster, de los recursos en ejecución, e información adicional que asegura el buen funcionamiento del cluster).

NODO DE LOS SERVICIOS DE ADMINISTRACIÓN

Se encarga de alojar los componentes administrativos que proveen APIs REST para que los usuarios finales puedan administrar la base de datos Azure SQL. Dichos servicios se ejecutan como servicios sin estado en múltiples nodos de administración activos. Si un nodo falla, el servicio se reinicia en otro y la operación continúa a partir del punto en donde se dio el fallo.

NODO DEL SERVICIO DE REDIRECCIÓN

Este nodo permite la redirección TDS a la base de datos Azure SQL.

La arquitectura utiliza además otros servicios para lograr una alta disponibilidad, balance de carga y gobernación de los recursos.

## ESLABONES DE SERVICIO DE LA BASE DE DATOS AZURE SQL

La base de datos Azure SQL está disponible en tres eslabones de servicio con múltiples niveles de rendimiento, disponibles bajo cada uno de ellos. Cada nivel de rendimiento contiene un conjunto creciente de recursos del sistema para proveer niveles crecientes de tasas de transferencia efectiva. Los recursos disponibles bajo cada nivel de rendimiento se expresan en términos de DTU (unidades de tasas de transferencia efectiva en la base de datos). Los DTU describen la cantidad relativa de potencia, memoria y tasas de E/S efectiva requeridas para completar una transacción de la base de datos.

## POOL ELÁSTICO DE LA BASE DE DATOS

La base de datos Azure SQL también permite la creación y la administración de varias bases de datos en un pool elástico de la base de datos. Éste provee un conjunto de DTUs compartidos o eDTUs asociados al pool, los cuales pueden ser utilizados por las bases de datos en el pool.

| ELASTIC POOL LIMITS | Basic | | | | | Standard | | | | | Premium | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| eDTUs per pool | 100 | 200 | 400 | 800 | 1,200 | 100 | 200 | 400 | 800 | 1,200 | 125 | 250 | 500 | 1,000 | 1,500 |
| Max. storage* | 10 GB | 20 GB | 39 GB | 78 GB | 117 GB | 100 GB | 200 GB | 400 GB | 800 GB | 1.2 TB | 63 GB | 125 GB | 250 GB | 500 GB | 750 GB |
| Max. DBs | 200 | 400 | | | | 200 | 400 | | | | 50 | | | | |
| Max. concurrent workers | 200 | 400 | 800 | 1,600 | 2,400 | 200 | 750 | 1,300 | 1,850 | 2,400 | 200 | 750 | 1,300 | 1,850 | 2,400 |
| Max. concurrent sessions | 2,400 | 4,800 | 9,600 | 19,200 | 28,800 | 2,400 | 4,800 | 9,600 | 19,200 | 28,800 | 2,400 | 4,800 | 9,600 | 19,200 | 28,800 |

| ELASTIC DB LIMITS | | | |
|---|---|---|---|
| Max. storage* | 2 GB | 250 GB | 500 GB |
| Min. eDTUs | 0, 5 | 0, 10, 20, 50, 100 | 0, 125, 250, 500, 1000 |
| Max. eDTUs | 5 | 10, 20, 50, 100 | 125, 250, 500, 1000 |

| BUSINESS CONTINUITY | | | |
|---|---|---|---|
| Point-in-time restore | Any point last 7 days | Any point last 14 days | Any point last 35 days |
| Disaster recovery | Geo-restore, restore to any Azure region | Standard geo-replication, offline secondary | Active geo-replication, up to four readable secondary backups |

*Databases share pool storage, so database storage is limited to the smaller of remaining pool storage or max. storage per database.

## ESLABONES DE SERVICIO: CAPACIDADES Y LÍMITES

- Tamaño máximo de la base de datos: Especifica el límite máximo para el tamaño de la base de datos.
- Respaldo automático y restauración de punto en el tiempo: Azure SQL provee características para el respaldo automático. Estos respaldos pueden ser restaurados en cualquier punto del tiempo dentro del período de retención especificado por los eslabones de servicio.
- Almacenamiento máximo de OLTP en memoria: Especifica el límite máximo de almacenamiento para el guardado de objetos OLTP en

memoria. Su desventaja es que es solamente aplicable a los eslabones de alta categoría.

- Máximo de solicitudes concurrentes: Es el número máximo de solicitudes concurrentes por parte del usuario o la aplicación que se pueden ejecutar sobre la base de datos.
- Máximo de inicios de sesión concurrentes: Representa el límite en el número de usuarios o aplicaciones permitidos para ingresar a la base de datos de forma simultánea. Este límite no es aplicable al pool elástico de la base de datos.

## HERRAMIENTAS DE ADMINISTRACIÓN

La administración de la base de datos Azure SQL puede ser hecho a través de las siguientes herramientas:

- Azure Portal
- SQL Server Management Studio
- Herramientas de datos en SQL Server
- Utilidades de línea de comandos y APIs REST

### AZURE PORTAL

Azure Portal es una aplicación basada en web que tiene las capacidades de crear, eliminar, restaurar, y administrar la base de datos Azure SQL y su servidor lógico asociado. Además, posee las capacidades de monitorear el rendimiento de la base de datos, configurar la seguridad y alta escalabilidad, y la capacidad de modificar el eslabón de servicio de la base de datos.

### SQL SERVER MANAGEMENT STUDIO

Las bases de datos Azure SQL soportan tanto la autenticación de SQL como la autenticación de Windows. Cuando se va a conectar a una base de datos Azure, se debe especificar el nombre del servidor SQL lógico que es obtenido como parte de la creación de la base de datos.

Se puede utilizar SSMS para una gran cantidad de actividades dentro de la base de datos, tales como:

- Monitoreo y administración usando los eventos extendidos
- Creación, diseño y desarrollo de la base de datos
- Administrar la seguridad del servidor SQL.

### SQL SERVER DATA TOOLS

SSDT es una herramienta gratuita que es usada para construir bases de datos relacionales en SQL Server y Azure SQL, paquetes SSIS, modelos de datos SSAS y reportes SSRS. SSDT es un ambiente de desarrollo que se puede utilizar para diseñar la base de datos Azure SQL.

## UTILIDADES DE LINEA DE COMANDOS Y REST API

Las utilidades de línea de comandos como PowerShell pueden ser empleados para crear y administrar bases de datos Azure SQL. Además, Azure contiene un conjunto de REST APIs que también son utilizados.

## AZURE SQL VS SQL SERVER EN LA MÁQUINA VIRTUAL DE AZURE

Azure SQL funciona muy bien en escenarios donde es necesario la provisión y la administración de varias bases de datos, en donde toda la administración y los gastos de parcheo están a cargo del vendedor, el cual ayuda a las organizaciones y los usuarios concentrarse solamente en el diseño y deploy de la base de datos. Sin embargo, no provee características periféricas como la replicación, agente de SQL Server, entre otros.

El uso de SQL Server en una máquina virtual de Azure es la opción apropiada para casos donde una organización busca extender sus despliegues bajo premisa en la nube. Con esta ventaja, el equipo de tecnología de información de la organización tiene el control administrativo absoluto sobre las máquinas virtuales.

## REPLICACIÓN DE TRANSACCIONES

La replicación de transacciones es una característica de replicación de datos o de esquemas de SQL Server. Ésta permite a los usuarios configurar las bases de datos de Azure SQL como suscriptores. Durante la etapa inicial, la base de datos suscriptora se sincroniza haciendo uso de una copia de la base de datos del publicador. Después de este proceso, todos los cambios que se realicen en la base de datos primaria serán capturados por logRead.exe y almacenados en la base de datos de distribución.

La base de datos Azure SQL no podrá ser configurado como distribuidor o publicador para una configuración de replicación transaccional.

## Capítulo 9: Continuidad de los negocios y la seguridad con Azure SQL

La continuidad del negocio es el asegurar que las aplicaciones críticas son resilientes a caídas de servicio (planificadas o no) que podrían resultar en pérdidas temporales o permanentes de las funcionalidades de negocio. Su meta es el diseño y deploy del negocio crítico de manera que las caídas de servicio tendrán minimizado su impacto en el negocio. Sus puntos fuertes de discusión son:

- Objetivo del tiempo de recuperación: El máximo tiempo de caída permitido para la aplicación, en el cual el negocio puede sufrir pérdidas monetarias.
- Objetivo del punto de recuperación: La máxima cantidad permitida de pérdida de datos antes de estar disponible para la aplicación.
- Tiempo estimado de recuperación: La duración estimada para que la base de datos vuelva a estar disponible después de una recuperación o failover.

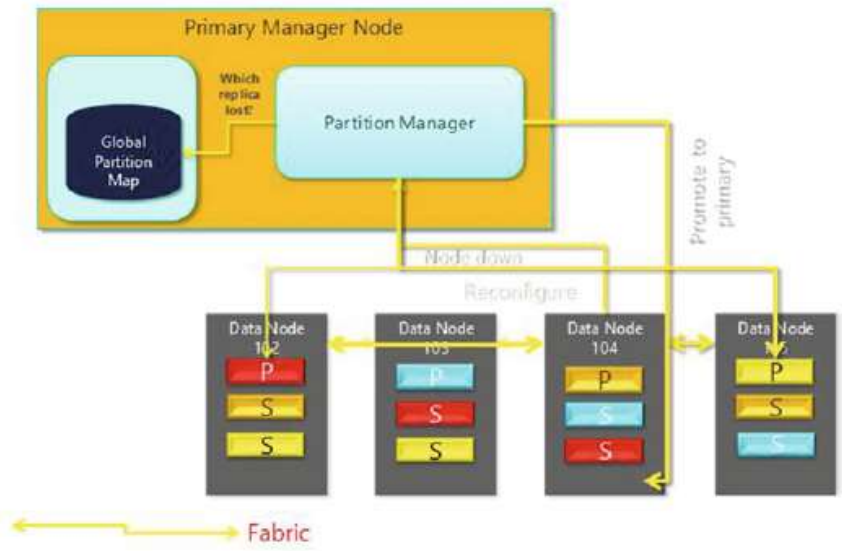
Cuando se diseña aplicaciones para la continuidad del negocio, deben considerar los tipos de caídas de servicio que provocan el fallo de la aplicación. Estos son los errores humanos, las caídas del sitio de información e incluso los procesos de actualización y mantenimiento.

### CONTINUIDAD DEL NEGOCIO Y RECUPERACIÓN POSTERIOR AL DESASTRE

#### REDUNDANCIA LOCAL

Por defecto, Azure SQL provee dos copias secundarias de la base de datos dentro del mismo centro de datos. Estas copias se sincronizan con la copia primaria de la base de datos, y además todas las operaciones de lectura/escritura se realizan sobre la copia primaria, para posteriormente ser replicada a las copias secundarias.

Azure presenta una copia lógica transparente de la base de datos para los usuarios finales. Si una de las copias falla, Azure verifica que otra copia es creada para mantener las tres copias de esa base de datos. Azure hace uso del administrador de particiones y del mapa de particiones globales para dicho fin, en cualquier punto del tiempo.




Lo que se ilustra en la figura anterior es un caso de fallo del nodo que contiene la copia primaria, en donde el administrador de particiones de Azure inicializa un algoritmo de failover en el cual una de las copias secundarias se convierte en una nueva copia primaria. Posteriormente, se repone la copia secundaria a través de la sincronización con su copia primaria.

RESTAURACIÓN EN UN PUNTO DEL TIEMPO

El servicio de Azure SQL DB provee capacidades de respaldo automático para todas las bases de datos. La retención de estos respaldos depende del eslabón de servicio en el que se ejecuta la base de datos. La base de datos puede ser restaurada en cualquier punto del tiempo durante el período de retención del respaldo.

Considere estos puntos clave respecto a la operación de restauración:

- Restore crea una nueva base de datos en el mismo servidor SQL lógico y con el eslabón de servicio, el cual fue el medio utilizado durante el punto de restauración.
- El tiempo de restauración depende de múltiples factores, tales como el tamaño de la base de datos, el punto de recuperación, el número de respaldos a ser restaurados, entre otros.
- Una vez que la base de datos es restaurada, será cargada al completo de acuerdo al eslabón de servicio y nivel de rendimiento usados.

Si una base de datos fue eliminada, el último respaldo de esa base de datos se retiene de acuerdo a las políticas de retención. Una base de datos eliminada puede restaurarse al punto en donde se dio el eliminado de la misma.

## GEORESTAURACIÓN

La funcionalidad GeoRestore de Azure permite que una base de datos pueda ser restaurada a partir de la copia georedundante. Azure realiza respaldo automatizado de la base de datos que va al almacenamiento georedundante, y GeoRestore utiliza esas copias con el fin de restaurar la BD en caso de algún desastre que impacte el sitio primario de información.

## GEOREPLICACIÓN

Provee la habilidad de crear réplicas secundarias geográficamente disparatadas de la base de datos primaria. A diferencia de las tecnologías de alta disponibilidad sobre premisa, como AlwaysOn en SQL Server, la georeplicación es por su naturaleza siempre asíncrona. Las transacciones en la copia primaria se envían a la segunda copia de forma asíncrona.

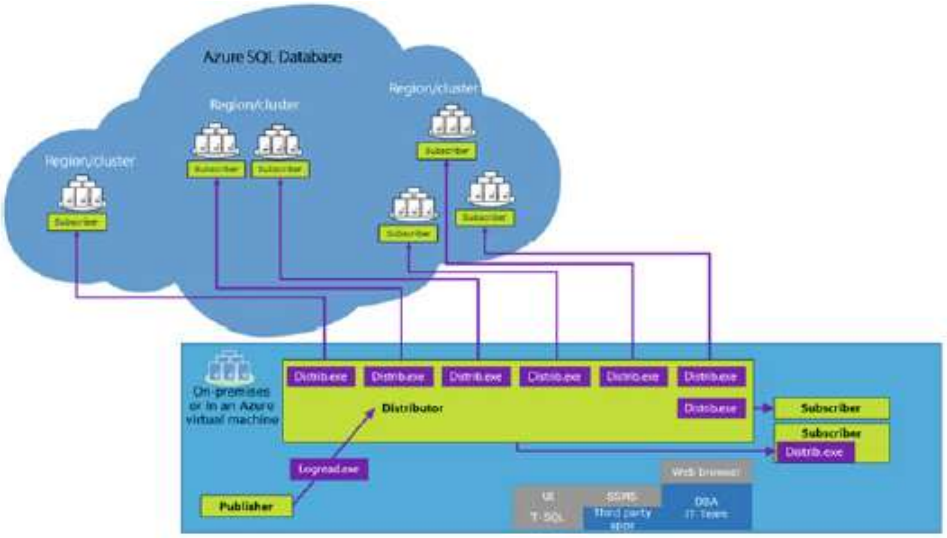
La georeplicación estándar crea una réplica secundaria sin atributos de lectura a una región DR Pair diseñada por Microsoft. Se requiere un failover manual para que sea accesible al usuario.

La georeplicación activa, mientras que se emplea la misma metodología de su contraparte estándar, crea réplicas secundarias con atributos de lectura, además de que cada centro de datos tiene capacidad para un máximo de cuatro réplicas secundarias, sin importar la región DR Pair a la que pertenece.

## REPLICACIÓN EN SQL SERVER

La replicación transaccional de SQL Server se da entre SQL Server sobre premisa o ejecutándose sobre una máquina virtual de Azure, y una base de datos Azure SQL. Existen consideraciones clave a tomar en cuenta a la hora de establecer el proceso de replicación entre el publicador de SQL Server y el suscriptor de Azure SQL:

- El publicador y el distribuidor deben ser ya sea instancias de SQL Server sobre premisa o de SQL Server sobre una máquina virtual de Azure.
- El suscriptor (base de datos de Azure SQL) debería estar en una suscripción push, lo que significa que el agente de distribución se ejecutará en el servidor de distribución.
- Todo el monitoreo y administración del proceso de replicación debe llevarse a cabo desde el servidor publicador.




AZURE SQL: SEGURIDAD Y AUDITORÍA

Las bases de datos en Azure SQL proveen un conjunto de características de seguridad para asegurar que los datos del usuario localizados en Azure no se vean afectados por ningún ataque. La seguridad multicapa disponible en Azure SQL, la cual incluye autorización basada en roles similar a la presente en SQL Server, además de la encriptación de datos en modo activo o en reposo, enmascarado de datos para restringir su acceso y la seguridad a nivel de filas, está diseñada para proveer protección completa contra cualquier amenaza real o percibida.

ADMINISTRACIÓN DEL FIREWALL

La primera capa de seguridad la provee el firewall de la base de datos de Azure SQL, el cual bloquea todas las conexiones no autorizadas a esa base de datos. A través del portal de Azure o de PowerShell se puede configurar la dirección IP o el rango de direcciones IP desde las cuales se les permite la conexión a la base de datos de Azure SQL, restringiendo de forma automática cualquier conexión que no esté en la lista blanca.

AUTENTICACIÓN Y AUTORIZACIÓN

Azure SQL soporta autenticación tanto de SQL como del Directorio Activo de Azure, además de autorización de acceso por roles de usuario. SQL provee roles a nivel de servidor y/o de base de datos, administrables a través del portal de Azure, línea de comandos en PowerShell o de SSMS.

SEGURIDAD A NIVEL DE FILAS

La seguridad a nivel de filas (RLS) controla el acceso a cada una de las filas de la tabla. Este control de acceso se implementa con predicados de seguridad en

la base de datos. Ya que la lógica del control de acceso está disponible en la base de datos, provee un mecanismo de seguridad muy confiable y robusto.

Ejemplo del uso de RLS sobre la tabla EmployeePerformanceData:

```
CREATE USER GeneralManager WITHOUT LOGIN;
CREATE USER Manager1 WITHOUT LOGIN;
CREATE USER Manager2 WITHOUT LOGIN;
CREATE TABLE EmployeePerformanceData
(
EmployeeID int,
EmployeeName varchar(200),
ManagerName sysname,
EmployeeRating int,
EmployeeIncrementPercent float
);

INSERT into EmployeePerformanceData values
(10, 'Employee10', 'Manager2', 1,10.00),
(11, 'Employee11', 'Manager2', 3, 6.53),
(12, 'Employee12', 'Manager1', 2, 8.71),
(13, 'Employee13', 'Manager2', 3, 6.25),
(14, 'Employee14', 'Manager1', 3, 5.87),
(15, 'Employee15', 'Manager2', 5, 0.00);
SELECT * FROM EmployeePerformanceData;
GRANT SELECT ON EmployeePerformanceData TO GeneralManager;
GRANT SELECT ON EmployeePerformanceData TO Manager1;
GRANT SELECT ON EmployeePerformanceData TO Manager2;

-- If any of the users select data from the table at this point, they
would see all 6 records
EXECUTE AS USER = 'GeneralManager';
SELECT * FROM EmployeePerformanceData;
REVERT;
EXECUTE AS USER = 'Manager1';
SELECT * FROM EmployeePerformanceData;
REVERT;
EXECUTE AS USER = 'Manager2';
SELECT * FROM EmployeePerformanceData;
REVERT;

--- Implement RLS using Security Predicates and Filters
/*

In this case we are creating a security predicate such that the
managers
can only their own Employee Data and the GM can see all the employee
information.
*/
CREATE SCHEMA Security;
GO
CREATE FUNCTION Security.fn_securitypredicate(@ManagerName AS sysname)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN SELECT 1 AS fn_securitypredicate_result
WHERE @ManagerName = USER_NAME() OR USER_NAME() = 'GeneralManager';

-- Tie the Security Predicate with the User Table
CREATE SECURITY POLICY SalesFilter
ADD FILTER PREDICATE Security.fn_securitypredicate(ManagerName)
ON dbo.EmployeePerformanceData
```

```
WITH (STATE = ON);

-- Now if we execute the Queries, each manager would only see their
own
employee information.
EXECUTE AS USER = 'GeneralManager';
SELECT * FROM EmployeePerformanceData;
REVERT;

EXECUTE AS USER = 'Manager1';
SELECT * FROM EmployeePerformanceData;
REVERT;
EXECUTE AS USER = 'Manager2';
SELECT * FROM EmployeePerformanceData;
REVERT;

-- Disable the security Policy
ALTER SECURITY POLICY SalesFilter
WITH (STATE = OFF);
```

## ENMASCARADO DINÁMICO DE DATOS

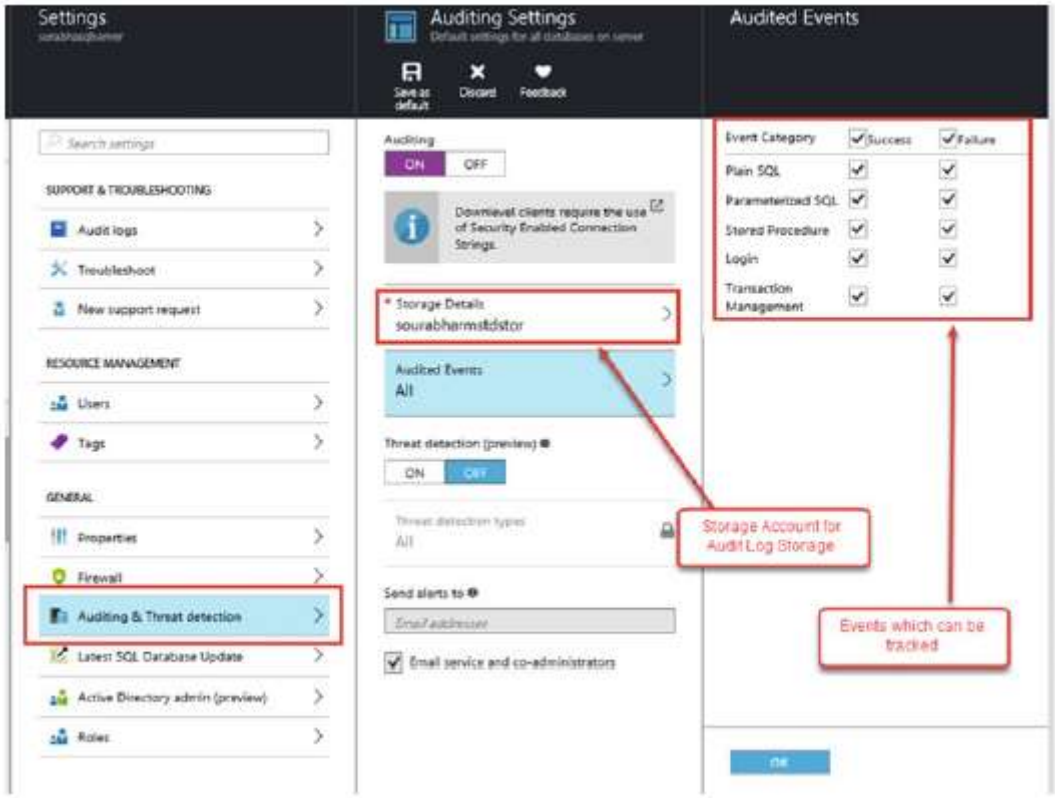
El enmascarado de datos previene la exposición y/o el acceso no autorizado a los datos más vulnerables antes de presentar la base de datos al usuario final. El enmascarado de datos se configura definiendo las políticas de seguridad dentro de la definición tabla/objeto y utilizando los permisos de enmascarado y desenmascarado para controlar la visibilidad de los datos. Los dueños y administradores de la base de datos tendrán acceso absoluto a los datos no enmascarados.

Ejemplo de enmascarado de datos:

```
CREATE TABLE Employee
(
EmployeeID int IDENTITY PRIMARY KEY,
FirstName varchar(100) MASKED WITH (FUNCTION =
'partial(1,"XXXXXXX",0)')
NULL,
LastName varchar(100) NOT NULL,
Phone# varchar(12) MASKED WITH (FUNCTION = 'default()') NULL,
Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL,
Salary float Masked with (Function='random(1,7)') Null
);
INSERT Employee (FirstName, LastName, Phone#, Email, Salary) VALUES
('Roberto', 'Tamburello', '555.123.4567', 'RTamburello@contoso.
com',100000.00),
('Janice', 'Galvin', '555.123.4568',
'JGalvin@contoso.com.co',200000.00),
('Zheng', 'Mu', '555.123.4569', 'ZMu@contoso.net',100000.00),
('Bill', 'Anderson', '555.123.4570', 'billand@contoso.net',150000.00),
('Graham', 'Scott', '555.123.4571',
'Grahamsco@contoso.net',120000.00);
SELECT * FROM Employee;
CREATE USER AppUser WITHOUT LOGIN;
GRANT SELECT ON Employee TO AppUser;
EXECUTE AS USER = 'AppUser';
SELECT * FROM Employee;
REVERT;
```

AUDITORÍA DE LA BASE DE DATOS SQL

Provee la habilidad de identificar los eventos clave dentro de la base de datos y los almacena en el servicio de almacenamiento de Azure. Estos pueden ser utilizados para requerimientos regulatorios y para proveer una prueba de rendimiento de las actividades en la base de datos.




DETECCIÓN DE AMENAZAS A LA BASE DE DATOS SQL

Provee un mecanismo para detectar y responder ante las amenazas potenciales (actividades anómalas) a la base de datos SQL. Usando una combinación de detección y auditoría de amenazas, los usuarios podrán investigar y tomar acciones necesarias ante cualquier amenaza a la base de datos.

CARACTERÍSTICAS DE ENCRIPTACIÓN

- ENCRIPTACIÓN DE CONEXIONES: Azure SQL permite a los usuarios hacer uso de una conexión SSL encriptada a la base de datos.
- ENCRIPTACIÓN TRANSPARENTE DE DATOS: Provee una forma de encriptar los datos en estado de reposo. Azure SQL usa la misma tecnología para lograr un encriptado de datos cuando la base de datos se encuentra en reposo. Se habilita con el código `Alter Database [MyDatabase1] set Encryption On`, a través del portal de Azure o línea de comandos en PowerShell.

- ENCRIPTACIÓN A NIVEL DE CELDAS: Este tipo de encriptación se logra con la combinación de llaves simétricas y las funciones de encriptado y desencriptado disponibles en SQL.
- SIEMPRE ENCRIPTADO: Es una nueva característica disponible a partir de Azure SQL y SQL Server 2016, y permite a los clientes administrar la encriptación de datos sensibles dentro de la aplicación, sin necesidad de revelar la lógica de encriptación a la capa de la base de datos. Esta característica soporta encriptación determinística y encriptación aleatoria. La determinística generará siempre el mismo resultado encriptado para el mismo texto, mientras que la aleatoria generará valores distintos cada vez que se va a encriptar.
  Ejemplo de una forma de uso:

```
USE [MyDatabase1]
GO
CREATE COLUMN MASTER KEY [CMKey]
WITH
(
KEY_STORE_PROVIDER_NAME = N'AZURE_KEY_VAULT',
KEY_PATH = N'https://<KeyVaultName>.vault.azure.net/keys/
AlwaysEncryptedkey'
)
GO
CREATE COLUMN ENCRYPTION KEY [CEKey]
WITH VALUES
(
COLUMN_MASTER_KEY = [CMKey],
      ALGORITHM = 'RSA_OAEP',
      ENCRYPTED_VALUE =

0x016E000001630075007200720065006E00740075007300650072002F006D00
79002F0034
006600330065003700660064003700360031003100330036003400340061003
6003100630
031003300640066003900370030006200320033006200370061003200620063
06600390
06600380066000DEF701B5FAB3F23266DADCAAE7B448122BA75BF1841DEF7143
A45C16D
37AA4AC57799D50596BA92C0406CC30A3D755D6F5D260DCCA42BB9926136985A
7CCF4537
B85330DA7C1B12047048A51B04A352F6C3E71BEFEAE777019506D11AC71AF8A7
AEC4DE7F
5B98ACF6EF7D56B0706E0D521533514335E500E476C6B1777212CE043BDD09B2
0BB97B5C7
31CB4D58BF8DDA38A7DF08EECE797DCC15A9E25B064003DE869F6D87B75A3F66
25A0162
92C3B8D8F8D3876DE62DDEE57F7BC2C901E3A2097B8E050862BEA0E33EF434D2
DED6D5F2
E54745D6E5C616932C5F2144B623C48B7643EDECE4CA545C31AB23DD2DFDF806
7D25C05E
F1786CCBC110E005D1567B53D6E34ACCC02052F6E9AE7365DE30856EF9DB5EC4
315770
D255FA76A9865204E8FBE5419AB5836480DE8345141073EB113E012CBF7132DC
C22
A3A32B6E44B961DDE2B0E7F24733062412CEF9C1A0DC96976A97D48EE5DCE4F5
AE12
13E680A31ADDFD9344A004ED59C6168CB7D5C8E42A22676A7D64F59A4C1687C6
1B5F603
```

```
49699A45D11B8EE7DC8DBB61A156AE70449483D93073497B23597A5F340A98FB
7BD37D9
DC926360E32F927BB672F6BE1FFC5C01760827AF24B603E184479905BA5DFA9C
23E523
182F7C5C8ABC53E5D6E6CB3806C5707EDBB7CAC3DE50DA4A2FC38D27EE65F263
8FFF
37483ABC1050EEAD835919B384BB9136C0F24A6BD9489910
)
GO


CREATE TABLE dbo.EncryptedTable
(
ID INT IDENTITY(1,1) PRIMARY KEY,
LastName NVARCHAR(32) COLLATE Latin1_General_BIN2
ENCRYPTED WITH
(
ENCRYPTION_TYPE = DETERMINISTIC,
ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
COLUMN_ENCRYPTION_KEY = [CEKey]
) NOT NULL,
Salary INT
ENCRYPTED WITH
(
ENCRYPTION_TYPE = RANDOMIZED,
ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
COLUMN_ENCRYPTION_KEY = [CEKey]
) NOT NULL
);
GO
```

**Conclusión:**

La base de datos SQL del servicio Azure ofrece características que favorece el desarrollador a la hora de programar una aplicación o sistema usando base de datos SQL, por ejemplo, algunas de las características son que, si su servicio necesita más recursos, se aprovisionará más recursos automáticamente. Si desea implementar base de datos adicionales, no ocupa realizar nada, además de la importancia de seleccionar la localización geográfica para mejorar el rendimiento.

Base de datos SQL usa una topología muy elaborado que maximiza el trabajo de redistribución, transparencia y recuperación. Una de las desventajas de usar la base de datos en la nube es que este se ejecuta en un ambiente multientorno si la aplicación emite una consulta grande que afecte a otras bases está conexión finaliza. Por eso debe tener en cuenta diversas consideraciones al diseñar una aplicación usando los servicios de Azure de la base de datos en la nube.

La explicación de un modelo altamente aceptado, CIA como se le conoce. Donde se cuenta con la integración de tres fases de forma en la que fortalece su modelo sobre le cuido de datos. Se explicaron tres métodos, tales como cifrado, hashing y certificado. Los cuales son diversos métodos que se pueden implementar para el cuido de datos. La parte de autenticación y autorización, las cuales son dos

subcategorías del control de acceso, las cuales se podrían considerar como el control que se ejerce sobre estos datos, donde se brinda una confianza acerca de que el manejo de los datos sensibles únicamente recae sobre personas autorizadas.

Microsoft brinda una facilidad en la conexión de una aplicación con la base de datos SQL, proporciona conexión para bibliotecas de clientes como ADO.NET, Java y PHP. La explicación de dos formas de conexión entre la base de datos y ADO.NET, así como con ODBC. La implementación de servicios de datos WCF donde se permite el consumo de servicios OData, así como su creación. Open Data Protocol el cual es un nuevo estándar de intercambio de datos el cual permitirá intercambio de datos entre diferentes sistemas.

Para la seguridad de las bases de datos, Azure posee características de seguridad tales como la georeplicación para restaurar la información en caso de fallo en el nodo primario del servidor, y las auditorías con el fin de protegerla frente a amenazas reales y/o percibidas, e incluso la posibilidad de controlar el acceso a una determinada fila de una tabla. También existe la encriptación total o parcial de los datos, sin importar si la base de datos está o no en plena ejecución, siendo la encriptación permanente la adición más reciente a las características que SQL Server ofrece.

**Bibliografía:**


- Klein, S. (2012). *Pro SQL Database for Windows Azure*. USA: Apress
- Mazumdar, P. (2016). *Pro SQL Server on Microsoft Azure*. USA: Apress
- Micosoft. (01 de Abril de 2016). *Walkthrough: Creating and Accessing a WCF Data Service in Visual Studio*. Microsoft Developer Network. Microsoft Recuperado de https://msdn.microsoft.com/es-es/library/cc668184.aspx